# LAPORAN RESMI
# PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK
# POLYMORPHISM


Fadilah Fahrul Hardiansyah S.ST., M. Kom

Ratri Maria Manik

3121600039

D4 TEKNIK INFORMATIKA – B

PROGRAM STUDI TEKNIK INFORMATIKA

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA


TA 2022/2023

## A. LATIHAN

1. Apa yang terjadi bila kode dibawah ini dikompile dan dijalankan jika sebelumnya Base.java belum dikompile? Jelaskan !

SuperclassX.java - Notepad

File Edit View

```
// Filename; SuperclassX.java
package packageX;

public class SuperclassX{
      int superclassVarX;


      protected void superclassMethodX(){
      }
}
```

SubclassY.java - Notepad

File Edit View

```
// Filename SubclassY.java
package packageX.packageY;

public class SubclassY extends SuperclassX{
      SuperclassX objX = new SubclassY();
      SubclassY objY = new SubclassY();

      void subclassMethodY(){
            objY.superclassMethodX();
            int i;
            i = objY.superclassVarX;
      }
}
```

```
C:\Users\ratri\OneDrive\Documents\OOP\Praktikum 13\packageX\packageY>javac SubclassY.java
SubclassY.java:4: error: cannot find symbol
public class SubclassY extends SuperclassX{
                               ^
  symbol: class SuperclassX
SubclassY.java:5: error: cannot find symbol
      SuperclassX objX = new SubclassY();
      ^
  symbol:   class SuperclassX
  location: class SubclassY
SubclassY.java:9: error: cannot find symbol
            objY.superclassMethodX();
                ^
  symbol:   method superclassMethodX()
  location: variable objY of type SubclassY
SubclassY.java:11: error: cannot find symbol
            i = objY.superclassVarX;
                     ^
  symbol:   variable superclassVarX
  location: variable objY of type SubclassY
4 errors
```

Analisa :

Terjadi error karena atribut pada pada parentclassnya memiliki modifier default, sehingga tidak bisa diakses oleh childclass.

2. Apa yang tampil di layar, jika kode dibawah ini dijalankan? Jelaskan!

Base.java - Notepad

File Edit View

```
class Base {
	int i = 99;

	Base(){
		amethod();
	}

	public void amethod(){
		System.out.println("Base.amethod()");
	}
}
```

Derived.java - Notepad

File Edit View

```
public class Derived extends Base{
	int i = -1;

	public static void main(String argv[]){
		Base b = new Derived();
		System.out.println(b.i);
		b.amethod();
	}

	public void amethod(){
		System.out.println("Derived.amethod()");
	}
}
```

```
C:\Users\ratri\OneDrive\Documents\OOP\Praktikum 13>javac Derived.java

C:\Users\ratri\OneDrive\Documents\OOP\Praktikum 13>java Derived
Derived.amethod()
99
Derived.amethod()

C:\Users\ratri\OneDrive\Documents\OOP\Praktikum 13>
```

Analisa :
Polimorfisme pada program terjadi pada **Base b = new Derived()** dimana terjadi instance objek b bertipe Base.

3. Apa yang tampil di layar, jika kode dibawah ini dijalankan? Jelaskan!

Parent.java - Notepad

File Edit View

```
class Parent{
	private void method1(){
		System.out.println("Parent's method1()");
	}

	public void method2(){
		System.out.println("Parent's method2()");
		method1();
	}
}
```

Child.java - Notepad

File Edit View

```
class Child extends Parent{
	public void method1(){
		System.out.println("Child's method1()");
	}

	public static void main(String args[]){
		Parent p = new Child();
		p.method2();
	}
}
```

```
C:\Users\ratri\OneDrive\Documents\OOP\Praktikum 13>javac Child.java

C:\Users\ratri\OneDrive\Documents\OOP\Praktikum 13>java Child
Parent's method2()
Parent's method1()

C:\Users\ratri\OneDrive\Documents\OOP\Praktikum 13>
```

Analisa :
Terjadi polimorfisme pada class Child di method main, terjadi instance objek p bertipe Parent lalu menjalankan constructor Child.

4. Suatu program terdiri dari class Pegawai sebagai parent class, class Manajer dan class Kurir sebagai subclass. Buatlah suatu program yang menerapkan konsep polymorphic argument sebagaimana yang telah disinggung dalam pembahasan sebelumnya.

Pegawai.java - Notepad

File Edit View

```
public class Pegawai{
	public String nama;

	public Pegawai(String nama){
		this.nama = nama;
	}

	public void info(Pegawai pegawai){
		if(pegawai instanceof Manajer){
			System.out.println(this.nama + " : Manajer");
		}

		else if(pegawai instanceof Kurir){
			System.out.println(this.nama + " : Kurir");
		}

		else{
			System.out.println(this.nama + " : Pegawai");
		}
	}

}
```

Manajer.java - Notepad

File Edit View

```
public class Manajer extends Pegawai{
	public Manajer(String nama){
		super(nama);
	}
}
```

Kurir.java - Notepad

File Edit View

```
public class Kurir extends Pegawai{
	public Kurir(String nama){
		super(nama);
	}
}
```

Test.java - Notepad

File Edit View

```
public class Test{
	public static void main(String args[]){
		Pegawai peg1 = new Pegawai("Fitri");
		Pegawai peg2 = new Manajer("Amelia");
		Pegawai peg3 = new Kurir("Nadra");

		peg1.info(peg1);
		peg2.info(peg2);
		peg3.info(peg3);
	}
}
```

```
C:\Users\ratri\OneDrive\Documents\OOP\Praktikum 13>javac Test.java

C:\Users\ratri\OneDrive\Documents\OOP\Praktikum 13>java Test
Fitri adalah Pegawai
Amelia adalah Manajer
Nadra adalah Kurir
```

## B. TUGAS

1. Pengimplementasian UML class Diagram

```
public class Pegawai{
	protected String nama;
	protected int gaji;

	public Pegawai(String nama, int gaji){
		this.nama = nama;
		this.gaji = gaji;
	}

	public int infoGaji(){
		return gaji;
	}
}
```

```
public class Manajer extends Pegawai{
	private int tunjangan;

	public Manajer(String nama, int gaji, int tunjangan){
		super(nama, gaji);
```

```
                this.tunjangan = tunjangan;
        }

        public int infoGaji(){
                return super.infoGaji();
        }

        public int infoTunjangan(){
                return tunjangan;
        }
}
```

```
public class Programmer extends Pegawai{
        private int bonus;

        public Programmer(String nama, int gaji, int bonus){
                super(nama, gaji);
                this.bonus = bonus;
        }

        public int infoGaji(){
                return super.infoGaji();
        }

        public int infoBonus(){
                return bonus;
        }
}
```

```
public class Bayaran{
        public int hitungbayaran(Pegawai peg){
                int uang = peg.infoGaji();
                if (peg instanceof Manajer)
                        uang += ((Manajer) peg).infoTunjangan();
                else if (peg instanceof Programmer)
                        uang += ((Programmer) peg).infoBonus();
                return uang;
        }

        public static void main(String args[]){
                Manajer man = new Manajer("Agus", 800, 50);
                Programmer prog = new Programmer("Budi", 600, 30);
                Bayaran hr = new Bayaran();

                System.out.println("Bayaran untuk Manajer : " + hr.hitungbayaran(man));
                System.out.println("Bayaran untuk Programmer : " +
hr.hitungbayaran(prog));

        }
}
```

```
C:\Users\ratri\OneDrive\Documents\OOP\Praktikum 13>javac Pegawai.java

C:\Users\ratri\OneDrive\Documents\OOP\Praktikum 13>javac Manajer.java

C:\Users\ratri\OneDrive\Documents\OOP\Praktikum 13>javac Programmer.java

C:\Users\ratri\OneDrive\Documents\OOP\Praktikum 13>javac Bayaran.java

C:\Users\ratri\OneDrive\Documents\OOP\Praktikum 13>java Bayaran
Bayaran untuk Manajer : 850
Bayaran untuk Programmer : 630

C:\Users\ratri\OneDrive\Documents\OOP\Praktikum 13>
```

Analisa :

Terjadi polimorfisme pada class Bayaran. Terdapat instance dari class Manajer, Programmer, dan Bayaran. Terbentuk objek baru bernama hr lalu menjalankan method hitungBayaran yang mengirim instance dari class Manajer dan Programmer. Pada method ini terjadi pengecekan asal instance menggunakan instanceof lalu menjalankan sesuai instancenya. Pada method ini juga terdapat casting objek parameter dari tipe parameter ke tipe asal agar kita dapat menggunakan kedua method yang berasal dari subclassnya.